

Konzept zur Variantenbildung von Oberflächen in einer Multikanal-Umgebung

Michael Hitz
DHBW-Stuttgart
Paulinenstraße 50
70178 Stuttgart
hitz@dhw-stuttgart.de

Abstract: Mehr und mehr werden im Kontext größerer Unternehmen Funktionalitäten der Backendsysteme über webbasierte Anwendungen den unterschiedlichen Nutzergruppen der Systeme zur Verfügung gestellt. Der Zugriff erfolgt meist abhängig von der Nutzerrolle. Hierzu wurde häufig für jeden dieser *fachlichen Kanäle* ein eigenes Portal aufgebaut, welches dann per Inter-, Extra- oder Intranet zur Verfügung stand.

In vielen Fällen führte dies dazu, dass Anwendungen (Oberflächen und Abläufe) parallel für unterschiedliche fachliche Kanäle entwickelt wurden, die sich ggf. nur marginal voneinander unterschieden und dieselben Backendsysteme nutzten. Zudem rücken in den letzten Jahren vermehrt mobile Geräte (*technische Kanäle*) weiter in den Fokus, die wiederum eigene Anforderungen an die Oberflächen und Abläufe stellen und zum Aufbau weiterer Portale mit ähnlichem Inhalt führen.

Die resultierenden Redundanzen verursachen in Wartung und Weiterentwicklung hohe Kosten. Um schnell und kosteneffizient auf den Markt reagieren zu können, bedarf es einer Lösung, welche Funktionalitäten kanalübergreifend nutzbar macht und bei Änderungen oder Erweiterungen Redundanzen durch Bildung von Varianten vermeidet - eine adaptive, multikanalfähige Architektur.

Das eingereichte Papier umreißt kurz, wie eine solche Architektur aussehen kann und fokussiert dann auf die Fragestellung, wie Abläufe und Oberflächen beschrieben werden können, um zu einem Anwendungsdesign zu gelangen, das die Erweiterbarkeit durch Variantenbildung in webbasierten Frontendsystemen ermöglicht.

1 Einleitung

1.1 Motivation

Der Zuschnitt der webbasierten Frontend-Systeme¹ auf bestimmte Nutzergruppen (**fachliche Kanäle**) in den vergangenen Jahren führte häufig dazu, dass parallel unterschiedliche

¹Der Begriff (*webbasiertes*) *Frontend-System* wird hier als Summe aller Komponenten verwendet, die notwendig sind, eine Anwendung dem Nutzer zu präsentieren. Dies beinhaltet Oberflächen, Seitenfluss und Anwendungslogik, welche den Zugriff auf Prozesse in den Backendsystemen orchestriert. Die Backendsysteme werden dabei als bestandsführend betrachtet und stellen die eigentlichen Businessfunktionalitäten bereit - und sind damit nicht Teil des Frontend-Systems.

Portale gebaut wurden, die inhaltlich sehr ähnliche bis gleiche Anwendungen enthielten², die aber speziell für jede Nutzergruppe neu entwickelt wurden. Oberflächen und Anwendungsabläufe wurden mehrfach entwickelt, obwohl sie in anderen Portalen bereits als Variante verfügbar waren. Kam nun ein neuer Kanal hinzu, der Zugang zu den Systemen des Unternehmens benötigte, wurde ein weiteres Portal erstellt.

Die wachsende Verbreitung neuer Technologien wie mobiler Endgeräte (Smartphones und Tablets), stellt weitere Anforderungen an die Oberflächen und Anwendungsabläufe. So werden hier aufgrund der kleineren Darstellungsfläche üblicherweise viel weniger Daten erhoben, als dies bei Desktop-Browser-Anwendungen der Fall ist. In vielen Fällen sind auch die Prozesse vereinfacht, was in anderen Dialogen und ggf. auch Abläufen resultiert. Aus Systemsicht sind dies weitere Kanäle (**technische Kanäle**), die spezifisch bedient werden müssen. Um diese unterstützen zu können, wurden sie meist als zusätzliche Portale in die Systemlandschaft eingefügt.

Die durch diese Mehrfachentwicklung entstandenen Redundanzen verursachen hohe Kosten in der Wartung und Weiterentwicklung und die Gefahr von Inkonsistenzen zwischen den Varianten bei Änderungen ist sehr hoch.

Die Anstrengungen der Unternehmen, Entwicklungskosten zu minimieren, erfordern es, neue (fachliche und technische) Kanäle schnell und zu geringen Kosten zu integrieren, um eine Multikanalstrategie erfolgreich umsetzen zu können (vgl. z.B. [Gro03]).

Mit technischen Mitteln kann dies erreicht werden, indem bestehende Oberflächen und Anwendungsabläufe wiederverwendet werden. Dazu bedarf es einer Architektur, die bei der Erstellung von Anwendungen die Bildung von Varianten bestehender Lösungen vorsieht und forciert. Eine mögliche Lösung ist die Schaffung eines multikanalfähigen Systems³, in welchem Varianten in bestehende Anwendungen zur Unterstützung neuer Kanäle einfach integriert werden können. In [Hit13] haben wir erste Ansätze hierzu bereits vorgestellt, welche insbesondere auf der *Bildung von Varianten* bestehender Oberflächen und Abläufe fokussierten.

1.2 Untersuchte Fragestellungen

Das in [Hit13] dargestellte Promotionsvorhaben im Projekt Ianus⁴, welches in Zusammenarbeit mit der DHBW-Stuttgart und einem deutschen Versicherer durchgeführt wird, befasst sich mit Konzepten zum Aufbau einer Multikanalarchitektur, wie sie Eingangs beschrieben wurde.

²z.B. ein Tarifierungsrechner für eine Krankenversicherung, der sowohl von Endkunden über das Internet als auch vom Vertreter bei einem Kundenbesuch verwendet werden kann. Der Grundablauf ist in beiden Fällen gleich. Die Variante des Vertreters könnte jedoch zusätzlich Gesundheitsfragen beinhalten, die in einem persönlichen Gespräch gestellt und erfasst werden können und zu einer genaueren Berechnung des Tarifs beitragen.

³Im Rahmen dieser Arbeit wird unter dem Begriff *Multikanalsystem* ein System verstanden, das in der Lage ist, unterschiedliche fachliche und technische Kanäle einheitlich zu bedienen, indem durch dessen Aufbau ein hoher Grad an Wiederverwendung erreicht wird und trotzdem die Spezifika der einzelnen Kanäle abgebildet werden können.

⁴Teilprojekt im Rahmen des Projektes "Kompetenzzentrums Open Source" an der DHBW-Stuttgart

Die Fragestellungen, die dort behandelt werden sollen, sind

- Welche Anforderungen leiten sich aus der Forderung nach Multikanalfähigkeit eines Systems ab und wie lassen sich diese Anforderungen in einem Basismodell einer Architektur erfüllen?
- Wie können Abläufe/Workflows multikanalfähig definiert und durch Variantenbildung die Spezifika weiterer Kanäle berücksichtigt werden?
- Wie können kanalspezifische Oberflächen in einem Multikanalsystem für diese Abläufe erstellt werden, sodass schnell und kostengünstig Erweiterungen erfolgen können?

Das vorliegende Papier beschreibt erste Ansätze zur Lösung insbesondere der letzten beiden Fragestellungen und befasst sich mit Fragen im Umfeld der Variantenbildung von Abläufen und Oberflächen, die im Rahmen einer Anwendung so erstellt werden sollen, dass sie einfach für weitere Kanäle wiederverwendet und für deren Erfordernisse erweitert werden können. Aufgrund der noch frühen Phase des Projekts sind viele dieser Fragestellungen noch nicht abschließend beantwortet und werden im weiteren Verlauf des Projekts näher untersucht und konkretisiert.

- Wie können Abläufe/Workflows gestaltet werden, dass sie die Bildung von Varianten unterstützen?
- Wie kann eine Oberflächen-Beschreibung erstellt werden, die in mehreren Technologien auf unterschiedlichen Geräten (technischen Kanälen) ausgegeben / „gerendert“ werden kann?
- Wie kann eine solche Beschreibung *kanalspezifisch* mehreren Geräten zur Verfügung gestellt werden?
- Wie ist eine kanalspezifische Seitenbeschreibung beschaffen?
- Kann eine kanalspezifische Seitenbeschreibung aus dem Datenmodell der Anwendung automatisch hergeleitet werden, wenn weitere Metainformationen zur Verfügung stehen?
- Wie können diese Metainformationen gestaltet werden und ist es möglich, diese in einem einzigen Modell abzubilden?
- Können die Metainformationen datenbezogen formuliert werden - also nicht darstellungsspezifisch - und trotzdem zur Herleitung einer vollständigen graphischen Präsentation dienen?

Um den Kontext für diese Fragestellungen zu setzen, wird zunächst ein Basismodell für eine Multikanal-Architektur vorgestellt und das Grundkonzept zur Bildung von Applikationsvarianten dargestellt. Anschließend wird das Konzept zur Variantenbildung auf der Ebene der Oberflächen dargestellt und dabei werden offene Fragestellungen aufgezeigt.

2 Konzept zur Herleitung von Oberflächen aus Modellen in einem Multikanalumfeld

2.1 Umfeld und Abgrenzung

Um ein möglichst einfaches und doch viele Anwendungsfälle umfassendes Szenario zu erhalten, wurde in Zusammenarbeit mit dem am Projekt beteiligten Versicherer vorab untersucht, welche Arten von Anwendungen häufig auftreten und wo ein Nutzen aus der Möglichkeit der Variantenbildung gezogen werden kann. Das Ergebnis war, dass insbesondere *datensammelnde Anwendungen* wie beispielsweise Tarifierungsrechner für Versicherungsprodukte oder kleine Dialogfolgen wie Adressänderung, Kontoänderung oder Überweisungen einen Großteil der Anwendungen in den Portalen darstellen und damit geeignete Kandidaten sind. Diese zeichnen sich dadurch aus, dass sie Dialogstrecken haben, die Daten sammeln und Punkte im Ablauf besitzen, an dem diese Daten verarbeitet werden (z.B. durch senden einer Anfrage an den Server und ggf. anschließende Weiterverarbeitung von Ergebnissen der Anfrage)⁵.

Als ein erstes Szenario aus dem Versicherungsbereich wurde das Thema „Tarifierung mit Antrag“ identifiziert, da es sich hier um geschlossene Prozesse handelt, die von mehreren Kanälen mit leichten Variationen durchgeführt werden.

Zudem wurde ein erstes Basismodell für eine Multikanalarchitektur hergeleitet, das als Grundlage für die weiteren Untersuchungen dienen konnte.

2.2 Basismodell einer Multikanalarchitektur⁶

Portale im heutigen Unternehmenskontext sind keine monolithischen Anwendungen „aus einer Hand“, sondern Aggregatoren von Inhalten (Content) und - meist kleinen - Funktionseinheiten (Applikationen), welche in den Content eingebettet sind und von unterschiedlichen Anbietern (z.B. IT-Bereichen) zur Verfügung gestellt werden.

Das hergeleitete Basismodell einer Multikanalarchitektur entstand unter der Annahme, dass mehrere fachliche Kanäle mit unterschiedlichen Geräten das webbasierte System nutzen. Das System soll für alle Kanäle einheitlich sein, sodass dieselbe Infrastruktur verwendet werden kann. Basierend auf den bisherigen Erfahrungen in der Portalentwicklung und der grundsätzlichen Entscheidung, einen serviceorientierten Ansatz bei der Integration der Backend-Systeme zu verwenden, ergab sich das in Abbildung 1 dargestellte Basismodell. Es folgt auf der Businesslogik-Ebene den in [BGK⁺97] angeführten Grundsätzen und

⁵Eine weitere Anwendungskategorie sind *datendarstellende Anwendungen*, welche hauptsächlich Daten aufbereiten und darsellen. Siese sind meist sehr speziell auf den Nutzer angepasst und weisen wenig Interaktion auf. Ein Beispiel hierfür wären Vertragsübersichten, die für unterschiedliche Kanäle sehr unterschiedlich ausfallen, da sich die darzustellenden Daten stark unterscheiden. Von hier findet häufig der Absprung in datensammelnde Anwendungen statt (z.B. vom Vertrag zur Änderung der Adresse des Versicherungsnehmers).

⁶Die Beschreibung des Basismodells in diesem Abschnitt ist weitgehend aus [Hit13] übernommen, wo bereits die ersten Ansätze zu dieser Arbeit und weitergehende Aufgabenstellungen im Umfeld skizziert wurden.

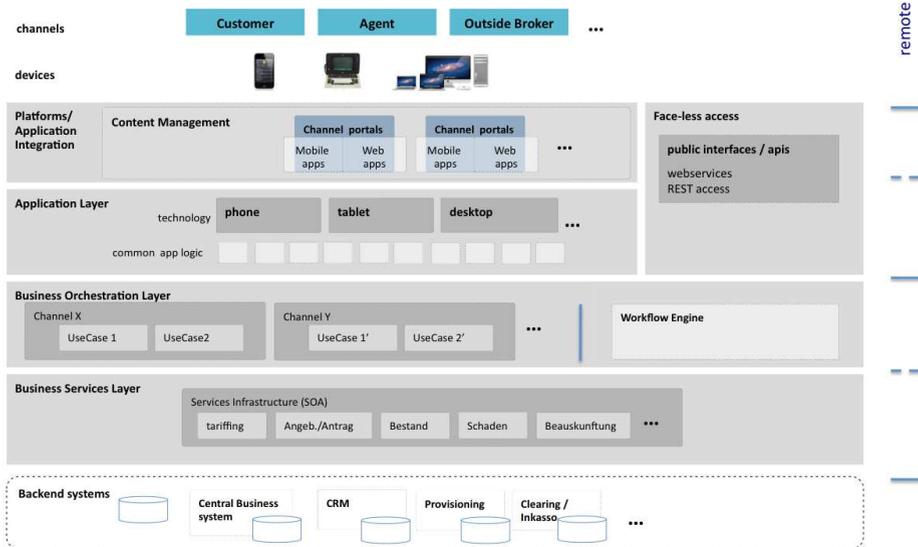


Abbildung 1: Basismodell für eine Multikanalarchitektur

konkretisiert die technischen Applikationsaspekte.

Den *kanalspezifischen Einstiegspunkt* bildet die mit *Platform/Application Integration* bezeichnete Ebene, die für die Bereitstellung und Aufbereitung kanalspezifischer Informationsinhalte und Integration der Anwendungen zuständig ist. Hierbei werden bestehende Anwendungen als Bausteine in den Content integriert (i.S. von JavaScript Mashups oder im Java Enterprise-Umfeld als Portlets).

Die Applikationsschicht (*Application Layer*) beinhaltet die Oberflächensteuerung der Anwendungen für unterschiedliche fachliche und technische Kanäle. Hier werden die Bausteine erstellt, die in unterschiedliche Portale integrierbar sein sollen und in ggf. unterschiedlichen Technologien erstellt sein können (Java Portlets, JavaScript-Anwendungen etc.).

Auf Ebene des *Business Orchestration Layer* befinden sich die Prozesse, die durch das Frontend orchestriert werden. Diese sind nach Anwendungsfall (z.B. „Tarifizierung einer Lebensversicherung“) und nach Kanal (z.B. „Außendienst“) gegliedert. Es ist sinnvoll, auf dieser Ebene *Business Process Engines* zu verwenden, da diese zu einfacher anpassbaren Systemen führen.

Hier werden Services des *Business Services Layers* genutzt, welche den Zugang zur eigentlichen Business-Logik und den Daten in den Backend-Systemen regeln.

Das gefundene Basismodell legt eine tiefere Untersuchung zur Variantenbildung insbesondere im *Application Layer* und *Business Orchestration Layer (BOL)* nahe, da hier kanalspezifischen Anpassungen häufiger vorkommen (vgl. auch [BGK⁺97]).

2.3 Varianten von Anwendungen

Das Ziel ist es, eine Anwendung zu erstellen, die in unterschiedlichen Portalen für unterschiedliche Kanäle eingesetzt werden kann. In [RPSO07] (S.23ff) werden Aspekte aufgeführt, die bei der Erstellung einer webbasierten Anwendung relevant sind und die Entwicklung beeinflussen. Hierbei werden auch Aspekte wie Performanz/Antwortverhalten, Sicherheit, Mehrsprachigkeit etc. aufgeführt. Viele dieser Aspekte erhalten einen noch höheren Stellenwert, wenn wir an Multikanalsystemen mit mehreren technischen Kanälen arbeiten, da dabei die Anforderungen mehrerer Kanäle berücksichtigt werden müssen. Dies bedeutet nicht zwangsläufig, einen Kompromiss finden zu müssen, sondern vielmehr mit der Vielgestaltigkeit umzugehen und dafür technische Lösungen anbieten zu können. Ein Beispiel hierfür ist die Technologie, mit der Oberflächen für Desktop- bzw. mobile Webanwendungen erstellt werden. Mobile Anwendungen haben andere technische Anforderungen und Restriktionen. Ein *one-size-fits-all*-Ansatz würde hier nur unbefriedigende Ergebnisse liefern.

Wir wollen im Folgenden die Aspekte

- Datenmodell,
- Oberfläche und
- Navigation

näher beleuchten und prüfen, wie sich die Bildung von Varianten im ausgewählten Anwendungsfall gestalten kann.

Zur Veranschaulichung der Überlegungen wird ein extrem vereinfachter Ablauf verwendet, der als Workflow in Abbildung 2 dargestellt ist. Der Ablauf stellt die Tarifierung eines Versicherungsproduktes dar. Hierzu werden in Schritten Daten vom Anwender abgefragt, die als Eingabe für eine Aktion (hier die Berechnung eines Tarifs für ein Produkt) benötigt werden. Das Ergebnis der Berechnung wird dem Anwender angezeigt und er kann entscheiden, ob er einen Antrag stellt oder ob er den Vorgang abbricht (der Abbruch ist nicht dargestellt)⁷.

Die Sequenz stellt eine Basisablauf dar, der für alle Kanäle (sowohl die Fachlichen als auch die Technischen) gleich ist.

In [Hit13] wurde skizziert, wie Varianten dieses Basisablaufs gebildet werden könnten, das hier konkretisiert werden soll⁸. Hierzu können im Grundablauf *Erweiterungspunkte (Extensionpoints)* vorgesehen werden, welche den Basisablauf um weitere kanalspezifische

⁷Die Darstellung als Sequenz und Workflow ist hier aus Gründen der Einfachheit gewählt. Es ist durchaus denkbar und sinnvoll, dass die datensammelnden Schritte (hier als Usertasks dargestellt) parallel an der Oberfläche abgefragt werden, indem dem Anwender beispielsweise Karteikartenreiter angezeigt werden, die er frei wählen und die Eingabefelder befüllen kann. Die Darstellung als Workflow bedeutet nicht, dass notwendigerweise ein Workflowsystem zum Einsatz kommen muss - deren Einsatz wird jedoch im Rahmen weiterer Arbeiten näher untersucht.

⁸Eine Ausarbeitung des Konzeptes wird ebenfalls in weiteren Arbeiten erfolgen. Im Rahmen dieses Papiers wird lediglich das Konzept dargestellt, um die Grundlagen für die Oberflächenüberlegungen zu schaffen.

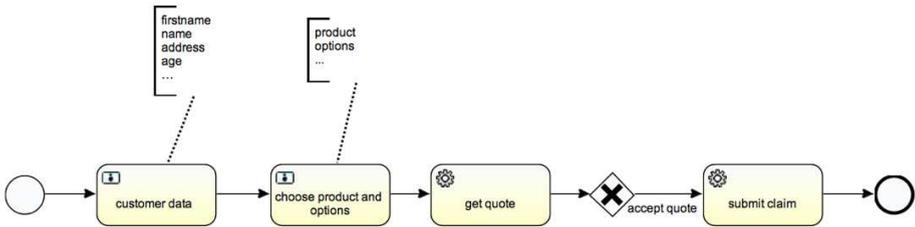


Abbildung 2: Basisablauf (stark vereinfacht)

Schritte/Unterabläufe erweitern. Diese Unterabläufe bestehen wiederum aus Schritten, die Daten sammeln und aus Aktionen auf den gesammelten Daten, die wiederum Ergebnisse liefern und den Ablauf beeinflussen können.

In Abbildung 3 ist dieser Ansatz dargestellt. An bestimmten Punkten im Ablauf sind Erweiterungspunkte vorgesehen. Die kanalspezifischen Unterabläufe könnten bei der Umsetzung mit einem Workflow-System in einem Verzeichnis (Registry) hinterlegt sein. Aus diesem kann basierend auf Informationen über den gerade den Ablauf rufenden Kanal der zu verwendende Unterablauf ermittelt werden. Im dargestellten Beispiel würden durch Nutzung der Anwendung durch einen Vertreter (Agent) am Anfang des Ablaufs die Vertreterdaten, bei Nutzung durch einen Makler (Broker) die Maklerdaten und bei Nutzung durch einen Endkunden keine weiteren Daten abgefragt.

Das **Datenmodell** der multikanalfähigen Anwendung besteht somit aus den im Basisablauf verwendeten Daten und der Menge aller Daten, die in den Unterabläufen ermittelt werden.

Der Umfang der abgefragten Daten variiert abhängig vom Kanal. So werden ggf. bei einem technischen Kanal *mobile device* in einem Schritt aus Platzgründen ggf. weniger Daten abgefragt als bei einem *desktop browser*. Dies gilt es sowohl in den Aktionen als auch den Oberflächen zu berücksichtigen.

Die **Oberflächen** in diesem Szenario sind Dialogmasken mit Ein-/Ausgabefeldern, welche mit den einzelnen Schritten korrespondieren. Je Schritt werden eine oder mehrere Masken benötigt.

Um das Ziel zu erreichen, *eine* Anwendung für alle Kanäle zur Verfügung zu stellen, müssen auch die Oberflächen Varianten zulassen. Es soll möglich sein, eine erstellte Oberfläche wieder zu verwenden und Varianten davon zu bilden. Schon die Forderung nach der Unterstützung unterschiedlicher Ausgabegeräte legt dabei eine abstrakte Beschreibung der Oberflächen nahe. Anforderungen an eine solche Beschreibung sind unter Anderem:

Die Beschreibung der Oberflächen muss **technologieneutral** erfolgen. Unterschiedliche technische Kanäle erfordern unterschiedliche Technologien zur Umsetzung. So könnte beispielsweise eine Desktop-Browser-Anwendung mit aktuell gängigen client-seitigen Frameworks wie *Sencha ExtJS*, *jQuery* oder *YahooUI* umgesetzt werden, während die mobile

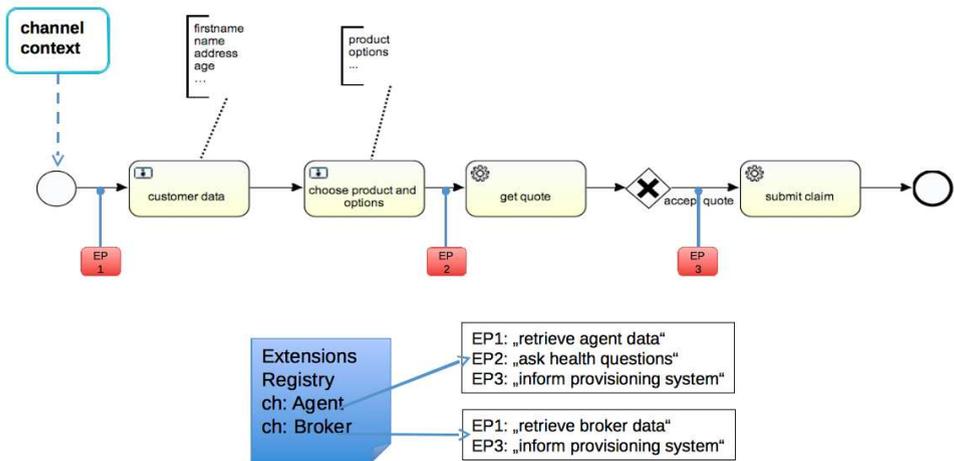


Abbildung 3: Basisablauf (stark vereinfacht)

Variante auf *jQuery mobile* oder *Sencha touch* setzt.

Die Daten, die in einem Schritt abgefragt werden, unterscheiden sich abhängig vom Kanal. Dies muss die Oberfläche berücksichtigen, indem die **Sichtbarkeit von Feldern kanal-abhängig** gesteuert werden kann.

Zudem muss berücksichtigt werden, dass auf unterschiedlichen Geräten bei größeren Datenmengen die Abfrage über mehrere Dialoge verteilt erfolgen muss. Es muss also abhängig vom Kanal eine **Verteilung der Daten** auf Seiten erfolgen können.

Die **Navigation** in einer multikanalfähigen Anwendung muss ebenfalls über Varianten flexibel gehalten werden. Neben der Hinzunahme oder dem Wegfall von Unterabläufen, die in der Navigation berücksichtigt werden müssen, kann je nach technischem Kanal auch die Art der Navigation variieren. So könnte in einer mobilen Anwendung die Navigation über reine Vorwärts-/Rückwärts-Navigation gelöst sein, um sequentiell alle Eingabeseiten zu bearbeiten und im Falle der Desktop-Browser-Anwendung die Navigation über eine Leiste/Menüstruktur erfolgen, über die wahlfrei auf die Eingabeseiten gesprungen werden kann.

Im Folgenden wird detaillierter auf Lösungsideen zur Beschreibung von Oberflächen eingegangen, die im Rahmen des Projektes aktuell verfolgt werden.

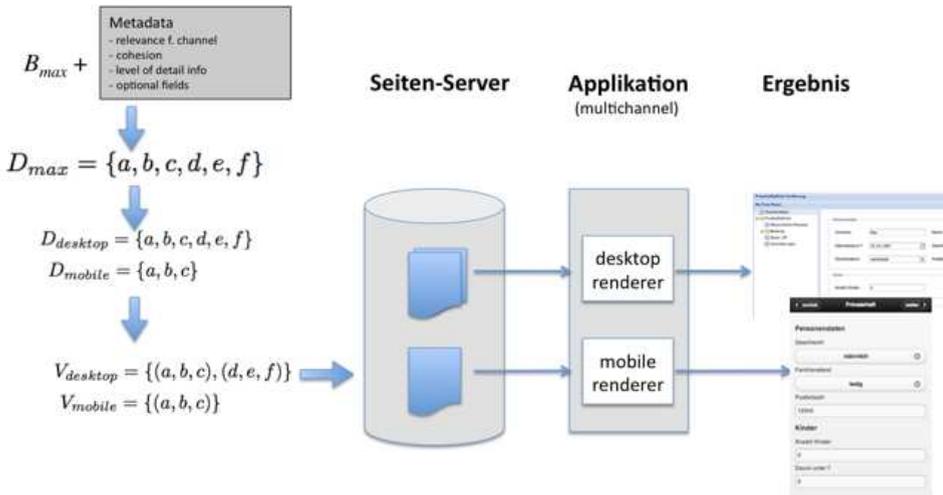


Abbildung 4: Herleitung von Oberflächen aus dem Datenmodell mit Metainformationen

2.4 Konzept: Herleitung variantenfähiger Oberflächen aus einfachen Datenmodellen mit Metainformationen

Das Ziel, einfach und schnell Varianten in einer Anwendung zu bauen, beinhaltet die Forderung, Varianten der Oberflächen erstellen zu können, die u.a. den oben gestellten Anforderungen nach Technologieneutralität, kanalabhängiger Sichtbarkeit von Feldern und Verteilung der Daten auf unterschiedliche Seiten abhängig vom nutzenden Kanal genügen. Die Beschreibung soll dabei einfach und leicht konsistent zu halten sein und dennoch nicht Aspekte vermischen (z.B. die Datensicht mit darstellungsspezifischen Informationen).

Der Ansatz, der im Rahmen unserer Arbeiten verfolgt wird, basiert darauf, das Datenmodell der Anwendung um Metainformationen zu erweitern, die es in einem weiteren Schritt gestatten, eine technologieneutrale aber kanalspezifische Seitenbeschreibung herzuleiten. Diese Seitenbeschreibung wird in einem letzten Schritt in eine Maske transformiert, welche für ein bestimmtes Gerät darstellbar und sinnvoll ist. In Abbildung 4 ist dies dargestellt.

Die Grundlage bildet die Grundmenge (B_{max}) der Daten, die im Rahmen des im letzten Abschnitt beschriebenen Basisablaufs inklusive aller Unterabläufe verarbeitet werden.

Diese Grunddaten werden jeweils mit Metainformationen versehen, welche die Daten und Beziehungen zu anderen Daten in der Menge näher beschreiben. Als Metainformationen können beispielsweise folgende Kriterien dienen:

- **Gruppierung von Daten.** Hierbei werden zusammengehörende Daten beschrieben. Ein Beispiel wäre *Straße*, *Hausnummer*, *PLZ* und *Ort*, die als Gruppe *Adresse* zusammengehören.

- **Kohäsion von Daten und Gruppen.** Dies bezeichnet den Zusammenhalt von Daten und Gruppen untereinander. So ist eine *Hausnummer* enger an *Straße* gebunden als *Ort*.
- **Relevanz über ein *level of detail* (LOD)-Konzept.** Um die explizite Angabe von Kanälen zu vermeiden, kann ein LOD-Konzept eingeführt werden, das Anwendungskategorien definiert, die nach steigendem Detailgrad immer mehr Daten abfragen. Ein konkreter Kanal muss dann nur noch mit einem Detailgrad assoziiert werden.
- **Relevanz für einen bestimmten Kanal bzw. Mengen.** Hier kann angezeigt werden, ob das Datum für einen bestimmten Kanal abgefragt werden soll. Es ist auch denkbar, die Daten Mengen zuzuordnen und über eine Festlegung, welcher Kanal welche Mengen darstellen soll zu einer feingranularen Zuordnung zu gelangen, die im Gegensatz zum LOD-Ansatz für die Kanäle disjunkt sein kann.
- **Typinformationen für ein Datum.** Typinformationen sind relevant bei der Darstellung. So könnten beispielsweise das Wissen um einen Typ *hausnummer* dahingehend genutzt werden, dass das Feld, in welchem der Typ abgefragt wird, nur 3 Zeichen lang sein darf und aus Zahlen plus einem Buchstaben bestehen darf.

In verwandten Arbeiten wie WebML ([CDM03]) (siehe Abschnitt „Arbeiten im Umfeld“) werden zur Modellierung dieser Aspekte meist unterschiedliche Modelle verwendet, die in Ihrer Gesamtheit das Endresultat herleitbar machen. Da dieser Ansatz die Gefahr von Inkonsistenzen birgt, soll im Rahmen dieser Arbeit möglichst *eine* Beschreibung der Zusammenhänge erfolgen. Im Rahmen z.T. schon laufender Arbeiten wird hierzu untersucht, welche Metainformationen notwendig sind, um am Ende eine korrekte grafischen Präsentation an der Oberfläche herleiten zu können.

Die Grunddaten der Menge B_{max} angereichert um die Metainformationen bilden in Abbildung 4 die Menge der Datenbeschreibungen D_{max} , auf denen die weiteren Schritte aufbauen. Die Elemente der Menge B_{max} ($a - f$) sind dabei die Kombination aus Datum und dessen Metainformationen.

In Abbildung 5 ist exemplarisch eine konkrete Repräsentation des Datenmodells mit Metainformationen ist dargestellt. Hier werden die zu sammelnden Daten gruppiert (1) und Daten/Gruppen mit Metadaten (2) versehen (z.B. *level-of-detail*, Typinformationen und Prüfvorschriften). Beispielhaft ist zudem eine enge Kohäsion dargestellt (3), die eine besondere Nähe von Daten zueinander darstellt.

Unterschiedliche Untermengen für die Kanäle können von D_{max} aufgrund der Metainformationen identifiziert werden. Die Kanaluordnung kann beispielsweise über den *level of detail* der Elemente erfolgen. So könnte eine mobile Anwendung auf den $level=1$ und eine Desktop-Browser-Anwendung auf den $level=4$ festgelegt werden. In Abbildung 4 wurden hierzu beispielhaft die technischen Kanäle *Desktop-Browser* und *mobile Anwendung* verwendet, die jeweils auf den hergeleiteten Teilmengen $D_{desktop}$ und D_{mobile} arbeiten.

Diese Untermengen werden in der Anwendung für bestimmte Kanäle auf unterschiedlichen Dialogseiten/Sichten gruppenweise abgefragt bzw. dargestellt ($V_{desktop}$, V_{mobile}).

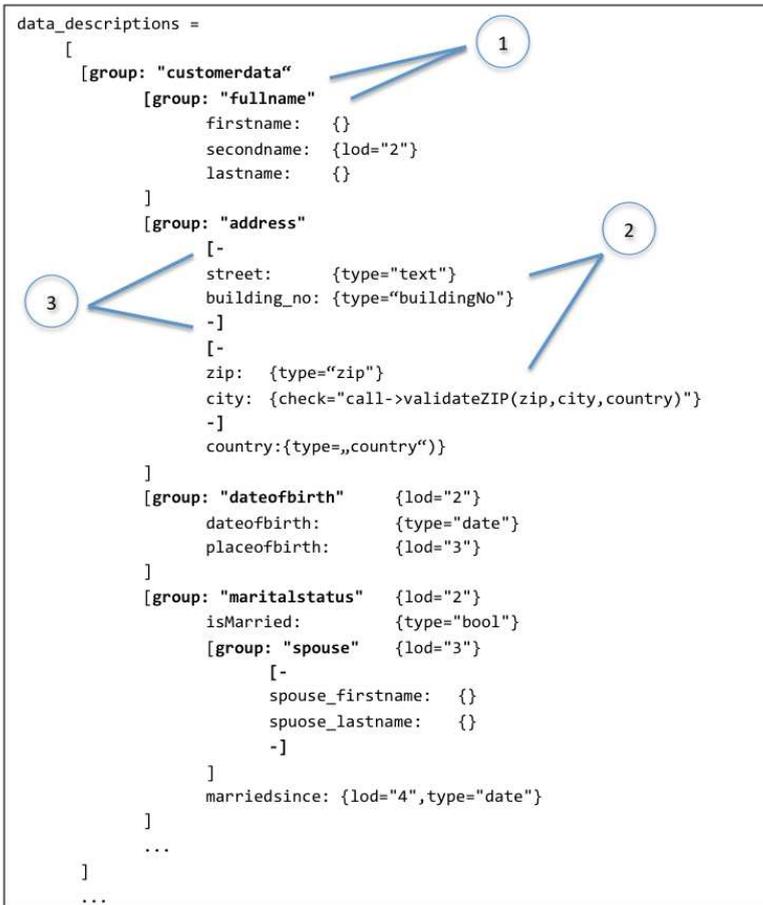


Abbildung 5: Exemplarische Beschreibung der Daten mit Metainformationen

Diese Verteilung auf unterschiedliche Seiten erfolgt basierend auf Regeln, die anhand der enthaltenen Metainformationen entscheiden müssen, wie die Daten verteilt werden können (z.B. über die Gruppierungs- und Kohäsionsinformationen von Gruppen in der Datenbeschreibung).

Die Regeln müssen gängigen Usability-Anforderungen genügen. Damit verbundene Fragen müssen in weiteren Arbeiten untersucht werden. Es wird jedoch davon ausgegangen, dass sich hierzu in der Literatur bereits weitreichende Ergebnisse im Bereich der Software-Ergonomie finden, auf die zurückgegriffen werden kann (z.B. [Hei11]).

Die Ergonomie bzw. Usability spielt aber auch innerhalb einer Seite eine Rolle z.B. bei der Anordnung der Felder. Ein triviales Beispiel ist die Darstellung der Felder *Straße* und *Hausnummer* bei einer Adressangabe. Diese Felder werden üblicherweise hintereinander

dargestellt. Dies ließe sich über eine starke Kohäsion der Daten beschreiben (Abb. 5, (3)). Üblicherweise ist das Eingabefeld für die Hausnummer zudem nur wenige Zeichen lang. Dies ließe sich aus der Typinformation über die Metadaten herleiten.

Das Resultat aus diesem Schritt ist eine **Seitenbeschreibung für einen spezifischen Kanal**, die jedoch noch technologieunabhängig ist. Der letzte Schritt besteht nun in einer Transformation dieser Beschreibungen in eine technologieabhängige Seitenbeschreibung, die auf dem Endgerät dargestellt werden kann.

Dieser letzte Schritt wurde im Rahmen des Projekts in Zusammenarbeit mit den Projektpartnern bereits prototypisch umgesetzt. Hierzu wurden die technologieneutralen Seitenbeschreibungen auf einem Seitenserver abgelegt. Eine Anwendung, die nun für ein bestimmtes Gerät erstellt und von einer bestimmten Nutzergruppe verwendet wird, erfragt die Seitenbeschreibung von diesem Server und verwendet die Beschreibung in einem zum Gerät passenden Renderer. Dieser erstellt aus der technologieneutralen Darstellung eine technologieabhängige Maskenbeschreibung, die dann auf dem Gerät ausgegeben werden kann. Im Rahmen des Prototyps wurde dies für drei Gerätetypen durchgeführt (*phone*, *tablet* und *desktop browser*), die jeweils client-seitig das Rendering der Seiten vornahmen. Hierbei kamen unterschiedliche Technologien zum Einsatz (*phone*: *jQuery mobile* + *Knockout*, *tablet*: *AngularJS* und *Twitter Bootstrap*, *desktop*: *Sencha ExtJS*), wodurch nachgewiesen werden konnte, dass technologie neutrale, kanalspezifische Seitenbeschreibungen für unterschiedliche Technologien verwendet werden können.

2.5 Arbeiten im Umfeld

Im Bereich der modellgetriebenen Entwicklung von Webanwendungen gibt es eine Reihe von Ansätzen in der Literatur, die hohe Relevanz für diese Arbeit besitzen.

WebML ([CFB⁺02]), OOHDM ([RS08]) und UWE ([KK03]) sind modellierende Ansätze für größere, navigationsintensive Anwendungen oder ganze Websites. Hierbei werden mehrere Modelle für unterschiedliche Aspekte des Systems erstellt und in einen Prozesskontext gesetzt. Teilweise werden in der Literatur auch Erweiterungen der Ansätze zum Bau von multikanalfähigen Anwendungen dargestellt (vgl. [CDM03]).

WebML ([CFB⁺02]) beispielsweise setzt den Fokus darauf, datenintensive Anwendungen und die damit verbundenen Aspekte zu beschreiben. Die Modellierung geht dabei bis hin zur Persistierung der verarbeiteten Daten. In Kombination mit den erstellten Modellen für die Oberflächen und Navigation kann dann eine vollständige Webanwendung generiert werden. Eine Trennung von Content und Anwendungsteilen, wie sie in einer Portalsicht angestrebt wird, steht hierbei nicht im Vordergrund, sondern die gesamtheitliche Modellierung einer komplexen, in sich geschlossenen Anwendung. Eine Multikanalfähigkeit kann in diesem Ansatz einfach erreicht werden, sofern es sich dabei lediglich um reduzierte Darstellungen handelt (vgl. [CDM03]). Werden Varianten benötigt, die eine andere Seiten- oder Navigationsstruktur aufweisen, müssen diese aufgrund des ganzheitlichen Ansatzes explizit modelliert werden.

Durch die parallele Erstellung mehrerer Modelle und deren Konsistenthaltung scheinen

diese Ansätze für die im Rahmen des Projektes betrachteten Anwendungsfälle jedoch zu mächtig. Sie fokussieren zudem nicht auf die Bildung von Varianten, wodurch ggf. Modellvarianten zu verwalten sind. Der hier vorgeschlagene Ansatz kann durch die Beschränkung der Anwendungsfälle über Konventionen vereinfachte Lösungswege aufzeigen. In zukünftigen Arbeiten muss jedoch detaillierter untersucht werden, welche in den vorhandenen modellgetriebenen Ansätzen enthaltenen Konzepte auch für die hier betrachteten Anwendungsfälle von Nutzen sind.

Einen für das Projekt relevanten Bezug haben auch existierende Ansätze im Umfeld der automatischen Generierung von Weboberflächen und der Navigation wie beispielsweise in [BGL06] und [YN08] und Beschreibung von Oberflächen, wie XForms ([DKMR03]) und XIML ([PE01]), die in weiteren Schritten noch näher betrachtet werden. Außerdem sind die Forschungen im Umfeld der *domain specific languages* relevant, welche die Basis für die Beschreibung des Datenmodells und der Metainformationen bilden.

In der Literatur finden sich eine Reihe von Arbeiten zur Variantenbildung in Workflows, welche den im Rahmen des Projektes erarbeiteten teilweise ähneln oder zumindest Grundlagen für die Umsetzung schaffen. In [WKNL07] werden Grundlagen für die Kontextsensitivität von Workflows gelegt und ein System entwickelt, welches basierend auf Kontext-Informationen agiert. Im Rahmen des ADEPT-Projektes (vgl. [DR09]) wird untersucht, wie Prozessversionen in ein System eingebracht werden können. Hierbei werden globale Modifikationen eines Prozesses betrachtet, die ggf. auf laufende Instanzen angewendet werden müssen, aber auch Möglichkeiten für ad hoc Modifikationen eines einzelnen laufenden Prozesses. Zudem finden sich hier Ansätze, welche die Konsistenz von definierten Prozessen sicherstellen. Die Konzepte zu den Prozessversionen zielen nicht auf die Integration von gleichzeitigen Varianten; jedoch ist das Thema Konsistenz von Varianten relevant, welches in diesen Arbeiten ebenfalls untersucht wurde.

Ein Ansatz, der sehr große Ähnlichkeit mit dem beschriebenen Extensionpoint-Ansatz aufweist, ist in [HBR08] dargestellt. Hier werden in der Prozessdefinition Marker platziert, zwischen denen die enthaltenen Knoten/Aktivitäten verändert werden können (z.B. löschen, einfügen, austauschen). Dieser Ansatz ist mächtiger, arbeitet jedoch aktuell nicht mit in BPMN standardisierten Mitteln und konnte so nicht im Projekt eingesetzt werden (Umsetzung nicht oder nur mit hohem Aufwand möglich). Der Extensionpoint-Mechanismus lässt sich einfach mit „Bordmitteln“ umsetzen und gestattet ein einfaches Management der Varianten über Konfigurationen.

3 Fazit / Ausblick

Wegen des Bausteincharakters heutiger Portalansätze, in welchem kleine, unabhängig entwickelte Funktionseinheiten zu einem Gesamtsystem aggregiert werden, und der Notwendigkeit zur Unterstützung unterschiedlicher Kanäle, begründet sich die Suche nach Möglichkeiten zur einfachen Erstellung multikanalfähiger Anwendungen. Der in diesem Papier vorgestellte Ansatz stellt speziell für die im Rahmen des Projekts betrachteten Anwendungsfälle eine leichtgewichtiger Alternative zu den in der gängigen Literatur vor-

handenen Ansätzen dar. Diese setzen den Schwerpunkt auf ein ganzheitliches Modellieren großer Webanwendungen und sind dadurch mächtiger, erfordern aber einen erhöhten Aufwand bei der Konsistenthaltung der einzelnen Modelle - insbesondere bei der Modellierung von Varianten, die dort explizit modelliert werden.

Durch die Fokussierung auf die datensammelnden Anwendungen, die in unterschiedliche Portale eingebettet werden können, verspricht der beschriebene Ansatz eine Vereinfachung der Entwicklung und Bildung von Varianten bestehender Anwendungen. Die Minimierung der zu modellierenden Aspekte und die dadurch übersichtliche Erweiterbarkeit unterstützt die einfache und kostengünstige Wartung und Erweiterung um hinzukommende Kanäle.

Aufgrund des frühen Stadiums der Arbeit bewegen sich die Konzepte allerdings noch auf einer hohen Abstraktionsebene. In weiteren Untersuchungen müssen zur Verifikation noch einige offene Fragen detaillierter betrachtet werden. Insbesondere müssen in der Literatur und Praxis bereits vorhandene Konzepte genauer analysiert werden, die aufgeführte Fragestellungen bereits lösen, um diese in die eigene Arbeit zu integrieren.

Es ist zu untersuchen, welche speziellen Darstellungselemente und -eigenschaften im Rahmen der betrachteten Anwendungskategorie benötigt werden, welche Metainformationen notwendig sind, um am Ende eine korrekte grafische Präsentation an der Oberfläche herleiten zu können, welche Beschreibungstechniken in bereits bestehenden Ansätzen existieren und wie diese zu einer einfach handhabbaren Lösung beitragen.

In einem weiteren Schritt ist zudem noch das Zusammenspiel des Konzeptes mit den umliegenden Komponenten auszuarbeiten - wie die Kombination der Oberflächen mit Workflows gestaltet werden kann, die beispielsweise über Workflowengines gesteuert werden.

Die Beschränkung der Anwendungsfälle wirft zudem die Frage nach den Grenzen des Konzeptes auf. Während der Ansatz für die Erstellung der unabhängigen Bausteine in einem Portal gut geeignet erscheint, muss betrachtet werden, ob das Konzept auch für komplexere Anwendungsfälle geeignet ist - indem diese beispielsweise auf kleinere Funktionseinheiten herunter gebrochen und damit wieder auf die beschriebenen Muster abgebildet werden können.

Literatur

- [BGK⁺97] Dirk Bäumer, Guido Gryczan, Rolf Knoll, Carola Lilienthal, Dirk Riehle und Heinz Züllighoven. Framework Development. *Communications of the ACM*, 40(10):52–59, 1997.
- [BGL06] Matthias Book, Volker Gruhn und Matthias Lehmann. Automatic dialog mask generation for device-independent web applications. In *Proceedings of the 6th international conference on Web engineering - ICWE '06*, Seiten 209–216, New York, USA, 2006. ACM Press.
- [CDM03] Stefano Ceri, Florian Daniel und Maristella Matera. Extending WebML for Modeling Multi-Channel Context-Aware Web Applications. In *Fourth International Conference*

on *Web Information Systems Engineering Workshops, 2003. Proceedings.*, Seiten 225–233, 2003.

- [CFB⁺02] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai und Mari-stella Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufman, 2002.
- [DKMR03] Micah Dubinko, Leigh Klotz, Roland Merrik und T. Raman. XForms 1.0 W3C Recommendation - <http://www.w3.org/TR/xforms>, 2003.
- [DR09] Peter Dadam und Manfred Reichert. The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science - Research and Development*, 23(2):81–97, April 2009.
- [Gro03] Sandra Christine Gronover. *Multi-Channel-Management*. Dissertation, Universität St. Gallen, 2003.
- [HBR08] Alena Hallerbach, Thomas Bauer und Manfred Reichert. Managing Process Variants in the Process Life Cycle. In *Proc. of the 10th Int. Conf. on Enterprise Information Systems*, 2008.
- [Hei11] Andreas Heinecke. *Mensch-Computer-Interaktion: Basiswissen für Entwickler und Gestalter*. Springer, 2011.
- [Hit13] Michael Hitz. Eine Multikanal-Architektur für Frontendsysteme und deren Erweiterbarkeit durch Variantenbildung. In *Proceedings, LNI, GI Software Engineering 2013, Workshopband*, Seiten 583–589. GI, Köllen Druck+Verlag GmbH, Bonn., 2013.
- [KK03] Nora Koch und Andreas Kraus. Towards a common metamodel for the design of web applications. In *Proceedings of the 3rd Intl Conference on Web Engineering (ICWE 2003), LNCE 2722*. Springer, 2003.
- [PE01] Angel Puerta und Jacob Eisenstein. XIML: A Universal Language for User Interfaces - <http://www.xml.org/documents/XimlWhitePaper.pdf>. *White Paper, RedWhale Software*, 2001.
- [RPSO07] Gustavo Rossi, Oscar Pastor, Daniel Schwabe und Luis Olsina. *Web Engineering: Modelling and Implementing Web Applications*. Springer, 2007.
- [RS08] Gustavo Rossi und Daniel Schwabe. Modeling and Implementing Web Applications with OOADM. In *Web Engineering: Modelling and Implementing Web Applications*, Kapitel 6. Springer, 2008.
- [WKNL07] Matthias Wieland, Oliver Kopp, Daniela Nicklas und Frank Leymann. Towards Context-aware Workflows. In *CAISE'07 Proceedings of the Workshop and Doctoral Consortium*, Seiten 1–15, 2007.
- [YN08] Takuto Yanagida und Hidetoshi Nonaka. Architecture for Migratory Adaptive User Interfaces. In *8th IEEE International Conference on Computer and Information Technology, 2008. CIT 2008.*, Seiten 450–455, 2008.